

Noise-contrastive estimation of normalising constants and GANs

Fonctions génériques

Algorithme d'Hasting

Utilité : simuler selon $p_m(., \psi)$ pour un paramètre ψ choisi.

Argument	Type	Exemple	Indication
x	vecteur	rcauchy(100, 0, 1)	notre échantillon de densité inconnue
n	entier	100	taille de la simulation
psi	vecteur	c(0,1)	paramètres de la fonction h
h	fonction		fonction qui retourne $\overline{p}_m(., \psi)$

```
hasting = function(x, n, psi, h){
  y = c()
  y = append(y, sample(sample(x, 1)))
  for (i in 2:n){
    y_ = rnorm(1, y[i-1], 1)
    u = runif(1)
    if ( u <=
          (h(y_,psi) * dnorm(y_, y[i-1], 1))
          / (h(y[i-1],psi) * dnorm(y[i-1], y_, 1))
    ){
      y = append(y, y_)
    }
    else {
      y = append(y, y[i-1])
    }
  }
  return (y)
}
```

Note : on peut très certainement écrire sous forme matricielle cette fonction pour une meilleure performance.

MC MLE

Utilité : retourne une estimation des paramètres selon la méthode décrite dans le papier de Geyer.

```
mc_mle = function(x, n, psi, h){

  y = hasting(x, n, psi, h)

  L = function(theta){
    return(sum(log(h(x,theta)/h(x,psi))) - n*log(mean(h(y,theta)/h(y,psi))))
  }

  theta = optim(
    par = rep(1,length(psi)),
    gr = "CG",
```

```

    control = list(fnscale=-1),
    fn = L
  )$par

  return(theta)
}

```

NCE

Utilité : Retourne l'estimation de la constante et des paramètres.

Argument	Type	Exemple	Indication
x	vecteur	rcauchy(100, 0, 1)	notre échantillon de densité inconnue
law_y	fonction	rnorm	fonction qui retourne un échantillon suivant la loi p_n
n	entier	100	taille de l'échantillon de bruit suivant la loi p_n
params_y	vecteur	c(0,1)	arguments de la fonction law_y
log_pm	fonction		fonction qui retourne le logarithme de la densité p_m
log_pn	fonction		fonction qui retourne le logarithme de la densité p_n
size_theta	entier	3	taille de θ , vaut habituellement 2 ou 3
method	string	"CG"	méthode d'optimisation, habituellement "CG" ou "BFGS"

```

nce = function(x, law_y, n, params_y, log_pm, log_pn, size_theta, methode = "CG"){

  y = do.call(law_y,c(list(n),params_y))

  m = length(x)

  h = function(u, theta){
    return( 1 / (1 + n/m * exp(log_pn(u) - log_pm(u, theta))))
  }

  J = function(theta){
    return( sum(log(h(x, theta))) + sum(log(1 - h(y, theta))) )
  }

  theta = optim(
    par = rep(1, size_theta),
    gr = methode,
    control = list(fnscale=-1),
    fn = J
  )$par

  return(c(theta[-1], exp(-theta[1])))
}

```

Graphiques

```

library(ggplot2)
library(reshape)

```

Utilité : afficher l'histogramme pour un échantillon de données x .

```
print_hist = function(x) {

  df = data.frame(x = x)

  hist_x = ggplot(df, aes(x=x)) + geom_histogram(aes(y = stat(count) / sum(count)), bins = 20, color="white", fill="black")

  print(hist_x)
}
```

Utilité : afficher l'évolution des paramètres au fur et à mesure de l'augmentation de n (la dimension de l'échantillon de bruit)

```
evolution_paramètres = function(x, law_y, params_y, log_pm, log_pn, nb_of_params, taille, precision, labels) {

  # Creation de l'abscisse
  m = length(x)
  N = seq(10, m*taille, length.out = precision)

  # Creation de l'ordonnée
  theta = c()
  for (n in N) {
    theta = append(theta, nce(x, law_y, n, params_y, log_pm, log_pn, nb_of_params, methode))
  }

  # Formatage des données
  theta = t(rbind(matrix(theta, nrow = nb_of_params), N))
  df = as.data.frame(theta)
  df_melted = melt(df, id.vars = 'N')

  # Plot
  plot_df = ggplot(df_melted, aes(x = N, y = value)) +
    geom_line(aes(color = variable, group = variable)) +
    geom_point(aes(color = variable, group = variable)) +
    labs(title = "Evolution des paramètres", x = "n", y = "Paramètres", color = "Légende") +
    scale_color_manual(labels = labels, values = c("blue", "red", "orange"))

  print(plot_df)

  return(theta)
}
```

Exemple basique : la loi normale

Soit x l'échantillon de taille m obtenu selon la loi de densité inconnue p_d .

On considère ici que p_d appartient à la famille de fonctions paramétrées par $\theta = (c, \mu, \sigma)$ suivante :

$$p_m(u; \theta) = \frac{1}{Z(\mu, \sigma)} \times \exp\left[-\frac{1}{2}\left(\frac{u - \mu}{\sigma}\right)^2\right] \quad \text{d'où} \quad \ln(p_m(u; \theta)) = c - \frac{1}{2}\left(\frac{u}{\sigma} - \frac{\mu}{\sigma}\right)^2$$

```
log_pm = function(u, theta){
  return(theta[1] - 1/2 * (u/theta[3] - theta[2]/theta[3]) ** 2)
  # theta[1] = c / theta[2] = mu / theta[3] = sigma
}
```

```
log_pn_cauchy = function(u){
  return(log(dcauchy(u, mean(x), sd(x))))
}

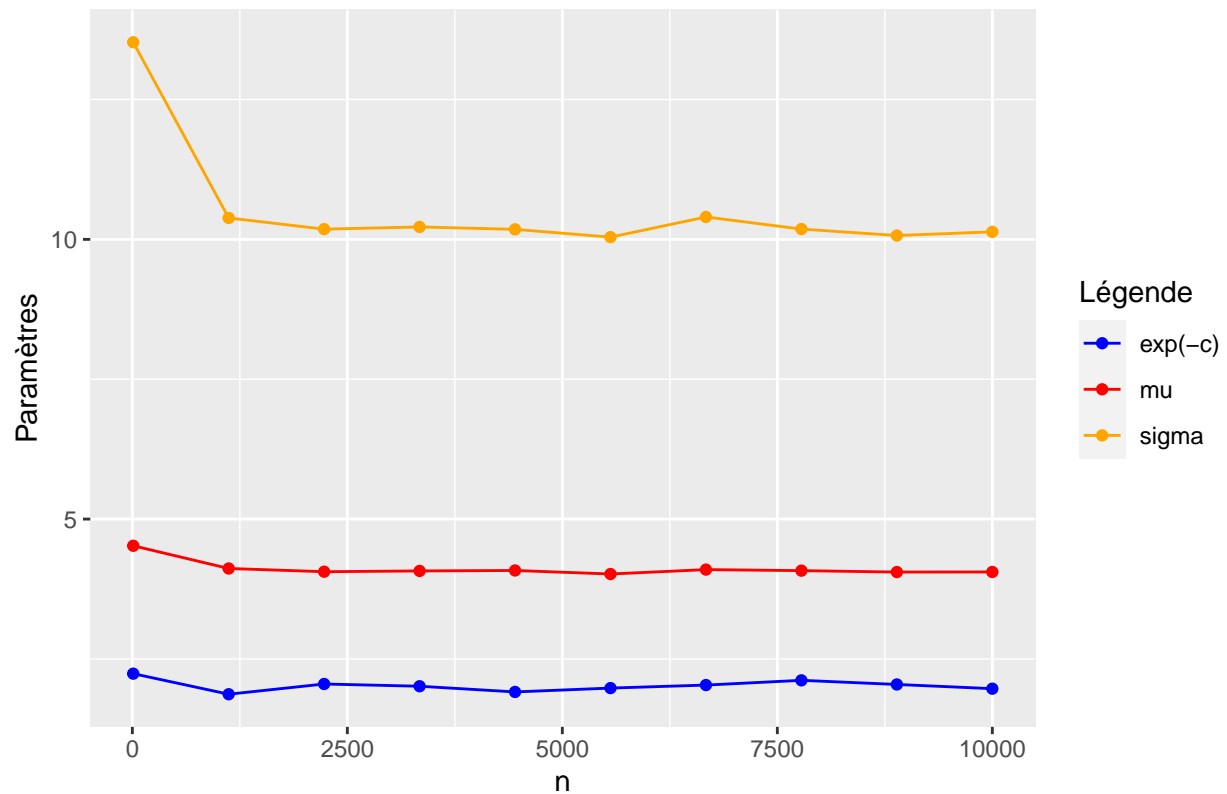
m = 1000
n = 10000
x = rnorm(m, 2, 4)
size_theta = 3

nce(x, rcauchy, n, c(mean(x),sd(x)), log_pm, log_pn_cauchy, size_theta)

## [1] 2.008353 4.107892 10.266842

evolution_paramètres(x, rcauchy, c(mean(x),sd(x)), log_pm, log_pn_cauchy, size_theta, 10, 10, c("exp(-c", "mu", "sigma"))
```

Evolution des paramètres



```
##
##      N
## [1,] 2.237046 4.523008 13.52282 10
## [2,] 1.869359 4.116812 10.38231 1120
## [3,] 2.052185 4.059414 10.18250 2230
## [4,] 2.012282 4.073139 10.22179 3340
## [5,] 1.910391 4.081459 10.17901 4450
## [6,] 1.979160 4.018137 10.03928 5560
## [7,] 2.033206 4.096970 10.40006 6670
## [8,] 2.117852 4.078202 10.18348 7780
## [9,] 2.043944 4.052974 10.06822 8890
## [10,] 1.968509 4.054899 10.13424 10000
```